More system independent usage of numerical verification algorithms written in high-level programming languages

OHLHUS, Kai Torben

Graduate School of Science
Tokyo Woman's Christian University

Workshop on Large-scale Parallel Numerical Computing Technology (LSPANC)
RIKEN Center for Computational Science (R-CCS), Kobe, Japan

January 29 – 30, 2020

Introduction

- **Verification methods / algorithms**:
  - "Mathematical theorems are formulated whose assumptions are verified with the aid of a computer." (Rump [7])
- **High-level programming language**:
  - Providing abstractions (e.g. no data types, memory management)
  - Less error prone, more expressive, faster development of algorithms.
  - Compiled or interpreted.
  - Not necessarily limited or slow. Depending on the purpose / computation.
  - **But:** Code / tools / libraries providing the abstraction become dependencies.

Introduction

- In my PhD thesis and before [6, 1] we computed rigorous error bounds for **conic linear programs** with up to **19 million variables and 27 thousand constraints**.

- The following simplified software stack (mostly high-level, interpreted code) was used[1]:

Define a conic problem $\mathcal{P} = (A, b, c, \mathcal{K})$ in either primal (P) or dual (D) standard form.

$$(P) \quad \begin{aligned} \min \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax = b, \\ & x \in \mathcal{K} \subset \mathbb{R}^n. \end{aligned}$$

$$(D) \quad \begin{aligned} \max \quad & \langle b, y \rangle \\ \text{s.t.} \quad & z := c - A^* y, \\ & z \in \mathcal{K}^* \subset \mathbb{R}^n, \\ & y \in \mathbb{R}^m. \end{aligned}$$

Compute an **approximate solution** $(\tilde{x}, \tilde{y}, \tilde{z})$ of $\mathcal{P}$ using a conic solver of your choice.

Compute **rigorous bounds** for the unknown optimal solution (x^*, y^*):

$\langle c, x^* \rangle \geq$ `rigorous_lower_bound` $(\mathcal{P}, \tilde{y})$
$\langle b, y^* \rangle \leq$ `rigorous_upper_bound` $(\mathcal{P}, \tilde{x})$

Optional a priori known upper bounds $(\overline{x}, \overline{y})$ on the optimal solution speed up the computation.

VSDP	
INTLAB	CSDP / MOSEK / SDPA / SeDuMi / SDPT3, ...
Matlab / GNU Octave	
"Linux"	

- For larger problem instances the current "Linux" system was insufficient.
  $\rightarrow$ Move to another "Linux" system, but...

[1] `https://vsdp.github.io/`

There is not just "Linux"...

First release	Distribution	Kernel	GCC[2]	Scilab	Octave	...
		5.5	9.2	6.0.2	5.1	
2019	RHEL/CentOS 8.1	4.18	8.3	-	-	
2018	SLES 15.1	4.12	7.2	-	-	
2018	Ubuntu 18.04.3	4.15	7.3	6.0.1	4.2	
2016	Ubuntu 16.04.6	4.4	5.4	5.5	4.0	
2014	SLES 12.4	4.12	4.8	-	-	
2014	RHEL/CentOS 7.7	3.10	4.8	-	3.8	
2010	RHEL/CentOS 6.10	2.6	4.4[3]	-	3.4	

[2] MATLAB® requires GCC 6.3
https://www.mathworks.com/support/requirements/supported-compilers.html

[3] No C11/C++11 support https://gcc.gnu.org/projects/cxx-status.html

What to expect from "Linux-clusters" today?

- Many **free and open source** high-level programming languages and libraries (e.g. Scilab, Octave, OpenBLAS, ...) are not suitable/not good performing:
  - outdated versions or missing
    - system dependent approaches (packages)
      (RedHat: devtoolsets, EPEL, ...; Ubuntu Backports)
    - system independent approaches
      (Anaconda [Python], flatpak, snap, ...)
  - configured for general purpose systems
    - linked against reference implementations

- Mostly **proprietary** pendants (e.g. MATLAB®, CUDA®, Intel®MKL, ...) are available in **more recent versions** on these "old" systems.

- Compiling missing software from source?
  - Dependencies often outdated too. In case of Octave: OpenBLAS, SuiteSparse, Arpack, ...
  - Space quotas, installation permissions, ...

- **Reproducibility** of previous results?

Sometimes things are even worse...

- Kashiwagi described a problem[4] for Linux + OpenBLAS (multiple threads) + Octave and switching of the directed rounding mode. It occurs in the following short example:

```
N = 5000;
A = rand(N);          % create random 5000x5000 matrix
B = rand(N);          %           elements in (0,1)
setround(-1);         % rounding downwards mode
C1 = A * B;           % lower bound for A*B
setround(+1);         % rounding upwards mode
C2 = A * B;           % upper bound for A*B
min(min(C2 - C1))     % should not be zero!
```

- The default OpenBLAS package of most Linux distributions is compiled using `CONSISTENT_FPCSR=0`, which means that the **floating-point control and status register** is not synchronized within multiple threads.
- The software stack relies on a **version** and a **configuration**.

[4] http://verifiedby.me/adiary/060

How to ensure complicated software stacks?

VSDP@2018	
INTLAB@11	CSDP@6.2.0 / MOSEK@8.1.0.62 / SDPA@7.3.8 / SeDuMi@1.32 / SDPT3@4.0, ...
GNU Octave@4.4.1 linked against OpenBLAS@0.3.7 configured with CONSISTENT_FPCSR=1 ...	

- Similar problem investigated by Shudler et al. [8] for SENSEI, presented on SC'19 in Denver.

Image: https://commons.wikimedia.org/wiki/File:Rock_balancing_(Counter_Balance).jpg

Supercomputing **pack**age manager[6] (1/5)

- Very actively developed since 2013 (started by members of the Lawrence Livermore National Laboratory).
- Contains currently 3838 packages (812 Python, 782 R).
- Addresses several problems with current Linux software distribution models:
  - Packages and (some!) dependencies are build from source. ($\neq$ ArchLinux).
  - Define target architecture, compiler (incl. icc), version, configuration (variants), ...
    $\rightarrow$ **Reproducibility!**
  - Packages **peacefully coexist** on the same machine.
    $\rightarrow$ **Maintenance!**
- Will be the default package manager for Fugaku[5].

[5] `https://postk-web.r-ccs.riken.jp/oss/public/`
[6] `https://spack.io` and [2].

Spack (2/5)

```
$ spack info openblas

Safe versions:
    0.3.7       https://github.com/xianyi/OpenBLAS/archive/v0.3.7.tar.gz
...
Variants:
    Name [Default]      Allowed values          Description
    ==============      ====================    ============================


    ilp64 [off]         True, False             Force 64-bit Fortran native
                                                integers
    pic [on]            True, False             Build position independent
                                                code
    shared [on]         True, False             Build shared libraries
    threads [none]      pthreads, openmp,       Multithreading support
                        none

$ spack install octave@5.1.0 ^openblas@0.3.7+ilp64 threads=openmp
```

CONSISTENT_FPCSR=1

Spack (3/5)

- Spack grammar in extended Backus-Naur form [2]:

$$
\begin{array}{lll}
\langle \textit{spec} \rangle & ::= & \langle \textit{id} \rangle \ [\ \langle \textit{constraints} \rangle\ ] \\
\langle \textit{constraints} \rangle & ::= & \{\ \text{'@'}\ \langle \textit{version-list} \rangle \mid \text{'+'}\ \langle \textit{variant} \rangle \\
 & & \mid \text{'-'}\ \langle \textit{variant} \rangle \quad \mid \text{'\textasciitilde'}\ \langle \textit{variant} \rangle \\
 & & \mid \text{'\%'}\ \langle \textit{compiler} \rangle \quad \mid \text{'='}\ \langle \textit{architecture} \rangle\ \} \\
 & & [\ \langle \textit{dep-list} \rangle\ ] \\
\langle \textit{dep-list} \rangle & ::= & \{\ \text{'\textasciicircum'}\ \langle \textit{spec} \rangle\ \} \\
\langle \textit{version-list} \rangle & ::= & \langle \textit{version} \rangle\ [\ \{\ \text{','}\ \langle \textit{version} \rangle\ \}\ ] \\
\langle \textit{version} \rangle & ::= & \langle \textit{id} \rangle \mid \langle \textit{id} \rangle\ \text{':'} \mid \text{':'}\ \langle \textit{id} \rangle \mid \langle \textit{id} \rangle\ \text{':'}\ \langle \textit{id} \rangle \\
\langle \textit{compiler} \rangle & ::= & \langle \textit{id} \rangle\ [\ \langle \textit{version-list} \rangle\ ] \\
\langle \textit{variant} \rangle & ::= & \langle \textit{id} \rangle \\
\langle \textit{architecture} \rangle & ::= & \langle \textit{id} \rangle \\
\langle \textit{id} \rangle & ::= & \texttt{[A-Za-z0-9\_][A-Za-z0-9\_.-]*}
\end{array}
$$

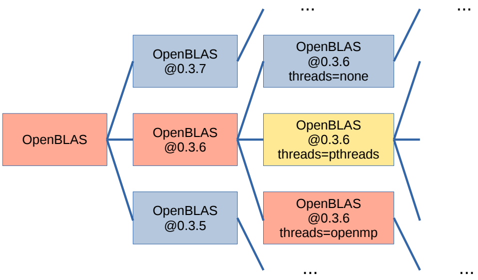
Spack (4/5)

- Example Spack package definition written in Python [2]:

```python
class Mpileaks(Package):
    """Tool to detect and report leaked MPI objects."""

    homepage = "https://github.com/hpc/mpileaks"
    url = homepage + "/releases/download/v1.0/mpileaks-1.0.tar.gz"

    version('1.0', '8838c574b39202a57d7c2d68692718aa')
    version('1.1', '4282eddb08ad8d36df15b06d4be38bcb')

    depends_on('mpi')
    depends_on('callpath')

    def install(self, spec, prefix):
        configure("--prefix=" + prefix,
                  "--with-callpath=" + spec['callpath'].prefix)
        make()
        make("install")
```

Spack (5/5) summary

- Spack allows to obtain more customized packages than classical Linux package managers.
- It addresses the resulting combinatorial configuration space by building only requested combinations.
- Build receipts are maintained and tested by several HPC facilities.



Singularity [4] - Container for HPC?

- Initial version created by Gregory M. Kurtzer about 2015 at the Berkeley National Lab.
- Still free software developed by Sylabs Inc.
- **Lightweight** container solution:
  - Container overhead negligible [3, 5, 9, 8].
  - Singularity images are a single self-contained file. Distribution by copy&paste, no DockerHub, ...
  - Native MPI, Ininiband, and GPU support.
  - No root daemon for the execution of Singularity images necessary. Runs with the privileges of the user. $\rightarrow$ Security.

Singularity definition files (1/3)

```
Bootstrap: docker
From: centos

%post

    # Install some development tools to build our code
    yum install -y \
      cmake \
      environment-modules \
      gcc-gfortran \
      gnuplot \
      python3 \
      texinfo \
      wget
```

Singularity definition files (2/3)

```
%post
    ...

    # Setup Spack
    cd /
    wget https://github.com/spack/spack/archive/develop.tar.gz
    tar -xf develop.tar.gz
    mv spack-develop spack
    source /spack/share/spack/setup-env.sh

    spack install octave@5.1.0 \
      ^ openblas@0.3.7+ilp64 threads=openmp \
                                 CONSISTENT_FPCSR=1

    # Tidy up, shrink container size ~710 MB --> ~590 MB

    rm -Rf /develop.tar.gz /spack/var/spack/cache/
    yum clean all
```

Singularity definition files (3/3)

```
%runscript

    # Commands to be executed, when container starts
    spack load octave
    octave

%environment

    export LC_ALL=en_US.UTF-8
    source /usr/share/Modules/init/sh
    source /spack/share/spack/setup-env.sh
```

Build and run Singularity Image Files (SIF)

- Build SIF with root privileges
  `sudo singularity build octave.sif octave.def`
  - Reduce image size, avoid unnecessary tools.
  - Reduce build time, trade-off install via Spack or guest Linux package manager. `yum` in this case.
- Run with user privileges
  `singularity run octave.sif`
  - Access to user's `/home` directory.
  - Other host system directories by default not accessible, but "`--bind`" possible.

- Summary
  - High-level programming languages provide useful abstractions for faster and less error prone development of verification methods.
  - To provide these abstractions sometimes nontrivial software stacks are required.
  - Spack can be used to uniquely specify and build these software stacks more independent of the underlying Linux distribution.
  - Singularity containers further increase independence of the underlying system without sacrificing security, InfiniBand, MPI, or CUDA support.
- Future work
  - Improve Spack receipts to support all required variations for VSDP.
  - More performance tests with Singularity containers and large scale linear conic programs.

Thank you for your attention!

Questions?

D. Chaykin et al. "Rigorous results in electronic structure calculations". In: (2016). URL: http://www.optimization-online.org/DB_HTML/2016/11/5730.html.

Todd Gamblin et al. "The Spack Package Manager: Bringing Order to HPC Software Chaos". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '15. Austin, Texas: Association for Computing Machinery, Nov. 15, 2015, pp. 1–12. DOI: 10.1145/2807591.2807623.

Stephen Herbein et al. "Resource Management for Running HPC Applications in Container Clouds". In: *High Performance Computing*. Ed. by Julian M. Kunkel, Pavan Balaji, and Jack Dongarra. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 261–278. ISBN: 978-3-319-41321-1. DOI: 10.1007/978-3-319-41321-1_14.

Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific Containers for Mobility of Compute". In: *PLOS ONE* 12.5 (May 11, 2017), e0177459. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0177459.

Emily Le and David Paz. “Performance Analysis of Applications Using Singularity Container on SDSC Comet”. In: *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. PEARC17. New Orleans, LA, USA: Association for Computing Machinery, July 9, 2017, pp. 1–4. ISBN: 978-1-4503-5272-7. DOI: 10.1145/3093338.3106737. URL: https://doi.org/10.1145/3093338.3106737.

Kai Torben Ohlhus. “Rigorose Fehlerschranken für das Elektronenstrukturproblem”. Thesis. Technische Universität Hamburg, 2019. DOI: 10.15480/882.2227.

Siegfried M. Rump. “Verification methods: Rigorous results using floating-point arithmetic”. In: *Acta Numerica* 19 (May 2010), pp. 287–449. ISSN: 1474-0508, 0962-4929. DOI: 10.1017/S096249291000005X.

Sergei Shudler et al. “Spack Meets Singularity: Creating Movable In-Situ Analysis Stacks with Ease”. In: *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. ISAV ’19. Denver, Colorado: ACM, 2019, pp. 34–38. ISBN: 978-1-4503-7723-2. DOI: 10.1145/3364228.3364682. URL: http://doi.acm.org/10.1145/3364228.3364682.

Andrew J. Younge et al. "A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds". In: *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Dec. 2017, pp. 74–81. DOI: 10.1109/CloudCom.2017.40.